The Thesis committee for Ana Isabel Guerrero de la Pena
Certifies that this is the approved version of the following thesis:

Computationally Efficient Path Planning Algorithm for Autonomous Navigation over Natural Terrain

APPROVED BY

SUPERVISING COMMITTEE:

Belinda Marchand, Supervisor

Wallace Fowler

Computationally Efficient Path Planning Algorithm for Autonomous Navigation over Natural Terrain

by

Ana Isabel Guerrero de la Pena, B.S.AsE

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

To my mother and her backpack

Acknowledgments

I would like to thank my advisor Dr. Belinda Marchand for guiding me, and for allowing me to continue developing this topic that I so much enjoyed. Thanks to Dr. Bishop for introducing me to this area of research, and encouraging me to continue through a graduate degree.

I wish to thank Dr. Fowler for acting as a second reader for this work, and for encouraging students to develop a true interest and passion in the field of aerospace engineering. It was while I was working on the undergraduate Mission Design project, a class taught by Dr. Fowler, that I began research on path planning for planetary exploration rovers.

I would like to thank my family, my mother, my father, and brother for all of their support; for never questioning but always encouraging my dreams and interests. To my parents, because I would not be here if it wasn't for them. I would like to thank my friends, who have been like a second family to me; and a special thanks to Hector, Bonnie, David, Jorge, and Ricardo, my classmates, lab partners and friends.

Computationally Efficient Path Planning Algorithm for Autonomous Navigation over Natural Terrain

Ana Isabel Guerrero de la Pena, M.S.E.
The University of Texas at Austin, 2011

Supervisor: Belinda Marchand

The present investigation focuses on the development of computationally efficient path planning algorithms for autonomous ground vehicles. The approach selected is based on a heuristic hill climbing local search. The cost index employed incorporates a traversability cost average, which offers two primary benefits: 1) the average extends the region of knowledge of the search algorithm, increasing optimality of the solution; and 2) the avoidance of hazardous regions is added to the decision making process. A binary traversability map representation is first utilized to analyze the performance of the enhanced heuristic hill climbing algorithm in comparison to the more traditional techniques. Next, the search algorithm is applied to a multi-valued traversability Map to test the capabilities of the algorithm over natural terrain. For this purpose, a digital elevation map is automatically processed to obtain multi-valued traversability values through the definition of a roughness, inclination and step

index. The complete path planning architecture for natural terrain then consists of a three step approach, computation of the multi-valued traversability map, implementation of the enhanced heuristic hill climbing search algorithm, and a path relaxation step. This last step is employed to fine-tune and smooth the trajectory, eliminating sharp turns caused by the regular characteristics of the search space.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In July of 1997, the Sojourner robotic rover, part of the payload of the Mars Pathfinder spacecraft, was the first vehicle of its kind to explore the surface of Mars. This initial mission was later followed by the Mars Exploration Rovers (MER), Spirit and Opportunity, each landing on different locations along the surface of Mars on January 2004. These rovers were specifically designed to travel farther than their predecessor. By 2009, the MER rovers had collectively traveled over 21 km, surpassing both the mark left by Sojourner and all designed expectations. Even though the MERs had on-board autonomous navigation and global path planning software, both the Sojourner and MER's were primarily controlled manually by Earth-based operators, and navigated with pre-planned commands based on visual images.

The computational requirements associated with autonomous path planning differs greatly relative to those of remote controlled vehicles. Earlier studies indicate that, in case of MER, remote operation was most efficient in contrast to autonomous navigation software.[2] Manual operation of the rovers required additional time since terrain hazards had to be identified from im-

agery first. Subsequently, a series of waypoints were defined by operators for the rover to follow.

For MER, the 26-minute communication delay introduced by remote manual operation of the rovers was deemed acceptable and did not hinder the mission goals in the end. However, if one considers a mobile system ona body farther away from the Earth, communication time delays may render remote operation infeasible. For example, Europa has been the subject of much interest among the scientific community. Due to Europa's close proximity to Jupiter, radiation concerns limit the lifetime of a robotic vehicle to less than two months.[3] A rover would have to traverse distances in orders of tens of kilometers in a short time, more than the MERs have done in years, with a higher delay than the one present on communications to Mars.

Studies suggest that the Mars Science Lab (MSL), NASA's next projected mission to land and operate a rover on Mars, would benefit from improvements in autonomous path planning, contact science, and drilling.[4] Once deployed onto the surface of Mars, the MSL rover is expected to travel more than 10 km,[5] for which it will require to look ahead and plan single waypoint trajectories of up to 50 m. This will require autonomous capabilities that reduce communication cycles from Earth if long range traverses are expected. For these reasons, it is necessary to improve autonomous capabilities on mobile robots for future planetary exploration missions.

This can be accomplished, in part, through improved terrain imagery sensors, enhanced computational capabilities, and overall design of mobile platforms. Certainly, improvement may be achieved through advanced navigation software that decreases the time and memory resources required by onboard path planning algorithms. This includes developing the technology used to image and represent the terrain, as well as the decision making process used to select safe and distance-efficient paths.

1.2 Research Contributions

Traditional path planning methods usually utilize a terrain map segmented as a regular grid and plan paths to a selected goal while avoiding obstacles. Most of these approaches restrict movement between grid cells strictly to the nodes in transitions of 90 or 45 degrees. Earlier studies[6–8] avoid such strict demands on the vehicle turning dynamics by introducing arc based movements, spline curves or interpolation. However, these methods add complexity to an otherwise simple grid search space by introducing a higher number of calculations for traversal difficulty and decision making

Apart from simple grid search spaces, traditional approaches assign a binary value to each grid cell representing an obstacle or free space. Approaches such as these account for obstacle presence, but disregard intrinsic terrain properties. Other authors assess the traversability characteristics of the terrain map using multi-valued representations of traversal difficulty for

each cell, and then use this information to compute a traversal cost function. In some cases[9] the quality of the terrain is represented with a Fuzzy Traversability Index chosen interactively by the user. Therefore, it requires a great amount of user input to define the traversability of map segments. This would deem such a method unfeasible for real-time applications on local terrain maps, where on-board sensors may dynamically detect new terrain features and the path planner require traversability indices to be computed on real-time.

Other investigations[1, 6, 7, 10–12] use methods that compute the traversal difficulty of map cells automatically. Among these, some authors[6, 10]compute the traversability of each cell by merging moment statistics and finding best-fit planes for a small rover-sized region. The plane parameters are used to compute hazard measures depending on step, roughness, pitch and border hazards. This method requires initial user analysis on four different parameters and their "hazard" thresholds. Two earlier studies[1, 11] use metrics such as pitch and roll of a plane and terrain roughness to determine the "goodness" of a cell; however, others[11] still categorize the traversability of each cell as "traversable" or "not traversable".

Most of the previously cited authors use a version of the A* or D* path planning search algorithms.[13] These types of best-first search algorithms try all possible extensions of a current path, and are therefore likely to find the global optimum if given enough time. Their implementation and optimality

is improved by decreasing map size in order to decrease computational time and memory for a single goal. This may not be desirable for longer traversal planning on real-time processing. These algorithms also use a simple cost function that usually avoids obstacles or hazard areas on a single cell basis, while prioritizing the optimization of path length.

1.3 Thesis Contribution and Organization

This thesis considers the problem of implementing a path planning algorithm that is capable of adding the avoidance of hazardous areas as a priority over short distance paths. Two cases are considered: 1) the terrain grid map is assigned binary valued traversability factors, 2) The terrain grid map is autonomously assigned multi-valued traversability factors. For the second case, the path planning architecture assumes that a Digital Elevation Map (DEM) is readily available as a point cloud map from a LIDAR (Light Detection and Ranging) sensor system; whereas for the first case, the Binary Traversability Map (BTM) is provided as an input. Therefore, the multi-valued Traversability Map (TM) architecture consists of an initial DEM Processing step. This step will provide the traversability factors required to construct the multi-valued map.

This effort proposes the following contributions to the path planning method

1. This work implements a two-step planning process where a preliminary

path is created over a simple grid search space, and then a path relaxation step optimizes the location of the path nodes, minimizing abrupt turning angles and path length. The first step utilizes a simple eight-connected grid search space that allows movement in transitions of 45 degrees. Though the type of motion allowed by such a node structure is not optimal for turning dynamics, it minimizes the number of nodes for decision making and cost computation of the path planner.

2. A heuristic Hill-climbing tree search algorithm is used to minimize a cost function and determine a path to the goal. Compared to the A* type search algorithms which try all possible extensions of a current path, the hill-climbing search produces a local optimum within the limited amount of time and memory available to perform a search, as is the case with real-time systems. Moreover, when combined with the average traversability cost algorithm, the knowledge of the local area is extended and the optimality of the node selected is enhanced.

3. A single parameter index is used to determine hazardous areas and traversability factors based on terrain roughness, inclination and step. This method requires an initial user analysis only on a single parameter and hazard threshold, and requires no further user input to determine traversal difficulty of specific areas.

4. The traversability factor is continuous-valued and autonomously computed from the DEM. This approach acknowledges intrinsic terrain prop-

erties such as slope and roughness

5. The cost function is not directly created by the traversability factor, rather by a traversability cost average. Decision making is based not on avoiding a single obstacle, but on preferring hazard-free areas for path planning.

1.3.1 Organization

This thesis is organized as follows:

- **Chapter 2. Background Chapter** A brief background on autonomous vehicles for planetary exploration is included. This chapter includes a brief knowledge covering from imaging sensors for terrain mapping to search algorithms and artificial intelligence (AI) applications for planetary exploration rovers.

- **Chapter 3. Mapping** This chapter explains the definition of the DEM and the local map. This map, on which the path planning algorithm will determine a safe traversal, is presented in detail. A brief description of the Global Map acquisition and interface with the local map is presented as well.

- **Chapter 4. Traversability Map Representation** The Traversability Map (TM) presented is a representation of the natural terrain that classifies difficulty of traversal for a mobile robot over areas, or cells, of

the local map. This chapter presents an algorithm to create the Multi-Valued Traversability Map given a DEM terrain representation.

- **Chapter 5. Path Planning** The first step of the trajectory planning process is explained in detail. The chapter discusses both the search space and the search algorithm used, along with the cost function used for optimization.

- **Chapter 6. Path Relaxation** This chapter presents a path relaxation algorithm that aims to optimize the path created by the initial path planning step. The relaxation algorithm fine tunes the location of the path nodes to minimize cost of the trajectory, as well as to minimize abrupt heading turns, creating a smooth path.

- **Chapter 7. Simulation and Results** Simulation parameters and corresponding results are shown. A BTM is utilized as an input to clearly demonstrate the improvements relevant to using the local search algorithm and cost function described. Secondly, results are shown for a multi-valued TM.

- **Chapter 8. Conclusions** The results of this research work are summarized. Suggestions are given for future work that may expand and improve the results here demonstrated.

Chapter 2

Background

Autonomous robotic systems may refer to autonomous manipulators for example, or to autonomous mobile robots. Autonomous manipulators, in the common form of a robotic arm, usually have a detailed knowledge of its environment and therefore path planning algorithms are based on a known map and goal. In contrast, autonomous mobile vehicles, in the common form of a rover with wheel dynamics, discover their environment as they traverse in search for a goal, which may not be initially fixed. Traditionally, maps traversed by exploration mobile rovers are referred to as partially known or dynamic, in the sense that the rover is dynamically imaging new terrain in previously occluded or out of sight areas as it moves. Therefore, the local map is constantly being updated and enhanced.

2.0.2 Terrain Perception

Traditionally, mobile rovers use stereo cameras to provide images of the terrain and create the map of the environment. A local map of the terrain can be maintained onboard by resampling and processing the range data generated by stereo vision. The MERs, for example, used this type of technology while

navigating on the surface of Mars. Stereo vision is an attractive technology for rover navigation because it is passive; sunlight provides all the energy needed for daylight operations, requiring only a small amount of power for environment imaging. One or two cameras can provide a wide enough field of view, so there is usually no need for moving parts in the system. Earlier studies[10] describe an algorithm to process and optimize stereo vision images for mobile rover navigation software in seven steps to create a Digital Elevation Map. A variety of image-processing operations, used to extract 3D information from camera vision, are available in the published literature.[13]

In recent years, LIDAR sensors have been proposed as an alternative imaging system to be used onboard autonomous exploration rovers. Laser range sensors are used extensively in 3D object recognition, 3D object modeling, and a wide variety of computer vision related fields. These sensors offer high-precision scanning abilities, with either single-flash scan or 360 degree scanning modes. LIDAR sensors eliminate the disparity problem inherent in the stereo camera systems by keeping the transmitted and received beams coaxial; and they can function despite of bad lighting conditions, making them useful for outdoor navigation.

LIDAR sensors output sets of points given in Polar coordinates,(θ, ϕ, ρ), which are easily transformed to (x,y,z) cartesian values. A cloud of points is created by either a single scan for a 3D flash LIDAR, or by varying the pitch angle of a LIDAR sensor producing a single horizontal beam. A cloud of

points is one of the multiple representations that can be produced from the raw sensors data, other representations include rectangular meshes, quad-tree representations and triangular grids.[1] A cloud point representation is known as the Digital Elevation Map, where a set of heights, z, corresponds to every coordinate pair (x,y). Thus, the DEM, useful for local rover navigation, can be easily obtained from a LIDAR imaging system compared to the multiple processing steps required for Stereo Camera imaging sensors.

A good perception of the environment is of utmost importance for navigation tasks. To move through the terrain to a selected goal, the rover must know where obstacles are located in order to avoid them. When traversing non-flat regions, a rover should be able to determine hazardous terrain that may cause it to fall or tip over as well as rough terrain that may add noise to on-board sensors. Therefore, after the terrain has been mapped, it is necessary to categorize it into traversable or untraversable regions. Terrain classification may depend on difficulty of traversal of a region as well as the expected precision and accuracy of the sensor readings. Methods that consider sensor error in terrain classification commonly enlarge obstacles to add a safe margin for traversal.

2.0.3 Approaches to Path Planning

Most approaches to path planning abstract the perceived environment map into a graph search space. The graph is then searched by some technique to determine a path to a goal.

Some path planners, for example, use Free Space Methods such as Voronoi diagrams, where corridors of empty space are defined between obstacles.[14] Paths are then allowed to run through these corridors only. Free Space methods suffer from extreme abstraction in narrow spaces and discard too much information on the environment.

Other path planners base their search on Vertex graphs, where paths are allowed through connected vertices if they do not intersect an obstacle. These methods often cause paths to cut through corners of objects, and similar to Free Space methods, also throw away information on the environment.

Potential Field approaches represent obstacles and hazardous terrain as hills with sloping sides such that the rover will stay away from them. The floor is seemingly tilted towards the goal to create a sink effect, however, regions in between obstacles will create local minimums in which the path planner can get stuck.

Regular Grid methods are a more traditional approach in which the terrain map is discretized as a regular grid of nodes connected to 4 or eight of its neighbors. Each node holds information on a cell of the map, stating whether the cell is traversable or not. Simple implementations restrict cells to be represented as either an obstacle or free space (a Binary Traversability Representation), and usually this restriction is placed by the method utilized to process the sensed images. The use of LIDAR sensors instead of stereo camera vision, producing a detailed representation of the terrain and easily converted to a DEM, allows for an easier creation of a flexible and multi-valued definition

of traversal difficulty.

Once a path planning approach has been defined, a search algorithm finds a solution to a goal as a set of steps over the state space. A search algorithm can be either uninformed, where states are expanded until a goal is satisfied, or informed, where there is an evaluation function that uses problem-specific knowledge to find the solution in a more efficient manner.

The most common search algorithm used for path planning applications is the A* search algorithm. The A* is a type of informed search algorithm which uses a "greedy" best-first search and a heuristic cost function. A greedy best-first search first expands the node with the best evaluation, this commonly being the one closer to the goal. A function used to estimate the evaluation cost as the distance to the goal is called a heuristic function.

This estimation is said to be heuristic because $h(\eta_{ij}) = 0$ if η_{ij} is the goal node. In the problem of finding a path for a traversing rover, the heuristic function is usually designed to be a straight line between the node η_{ij} and the goal node, giving an estimate of the solution cost. This heuristic cost is also optimistic because it does not overestimate the cost to reach the goal. Selecting an optimistic heuristic cost is important for the computational efficiency of the search method.[13] However, this cost is not by itself optimal nor complete, which is why the A* algorithm adds the "greedy" cost from the start node to the current node, $g(\eta_{ij})$ such that

$$f(\eta_{ij}) = g(\eta_{ij}) + h(\eta_{ij}) \tag{2.1}$$

where $f(\eta) =$ *estimated cost of the cheapest solution through* η

Using a heuristic cost in the function evaluation does not guarantee that the total cost will exhibit monotonicity along the path from the root node. Therefore, A* uses a modified cost function:

$$f(\eta') = min(f(\eta)g(\eta') + h(\eta')) \tag{2.2}$$

where the node η is the root of the node η'. A* then checks if $f(\eta')$ is less than $f(\eta)$ to make sure that f decreases along the path from that root. If f never decreases along the path, traditionally, the A* algorithm uses an incremental heuristic search to look for an alternate path that minimizes the current f cost, potentially expanding all nodes with a lower cost.

A* therefore presents a problem: the number of nodes expanded in search of the goal can have exponential growth. Because it keeps all generated nodes and their information in memory, A* usually runs out of memory space before it runs out of time. The A* search is both optimal and complete, however it may not be practical for implementation on large maps, which ultimately is required for long range traversals.

Now that the most common types of architectures utilized for path planning of mobile rovers have been described, the following chapters pro-

pose an improved architecture that encompasses the imaging method of the environment, the technique to process the terrain map and assign traversal difficulty over the search space, and a search algorithm that allows a practical computation efficiency, altogether returning an adequate path to the goal.

Chapter 3

Mapping

3.1 Mapping based on 3D sensing

Two representations of the surface with different levels of detail are introduced. One of them having a wider or global coverage of a given area with a lower resolution and a second one covering a smaller or local area with a higher resolution. The low resolution map is defined as a Global Map, while the higher resolution map will be defined as the Local Map (LM). The path planner will be executed over the Local Map in order to determine an optimal trajectory to a goal location. Therefore, the Local Map must have a high resolution sufficient for safe rover traversal.

3.1.1 Global Map

Satellite imagery can be used to create a global map with a resolution in the order of meters. This low resolution global map would be concerned with traversal to a feature of interest, avoiding extended hazardous areas. Mission operators could determine a single goal on the global map, and a global path planner can be implemented to create a guideline trajectory, segmented into waypoints.

Thus, orbital imagery is crucial for long-range planning, but cannot resolve vehicle hazards on the order of centimeters. Since the resolution of the global map is not detailed enough for safe rover traversal, the map must be segmented into local regions. The location of the waypoints and size of these Local Maps may be determined by the sensing range of the on-board terrain imaging system. Each waypoint would then become endpoints, or local goals, on the Local Map. Earlier studies[1, 9] describe a global path planner with a multi-valued traversability categorization of the terrain.

Segmenting a global map into local maps poses a navigation difficulty for an autonomous mobile system: matching the local maps to the global map with minimum positioning errors. Wheel odometry, visual odometry and dead reckoning are commonly used for rover navigation on a local map, however an alternative solution must be used to align a rover-based local map to a satellite-based global map. Carle, Furgale, and Barfoot[15] propose matching 3D LIDAR scans to features detected by a 3D orbital elevation map. The method described can be used to estimate the position of a rover on the LM with respect to the global map, improving long-range autonomous traverses. Although the global map is an important element in long-range path planning, the following sections will focus on the definition and determination of the Local Map as well as its use by the local path planning algorithm.

3.1.2 Digital Elevation Map

Data obtained from the surface by a sensor system on board of the vehicle can be gathered to build a local map in the manner of a Digital Elevation Map. A LIDAR based system, compared to stereo camera systems, is capable of performing under poor illumination conditions. LIDAR sensors also eliminate the disparity problem that arises by using stereo camera systems.

The output values obtained from the LIDAR sensor are sets of polar coordinates formed by the distance, horizontal and vertical angles (ρ, θ, ϕ) that define the location of a sensed point. In order to create a 3D point cloud, it is necessary to transform the polar data into sets of (x,y,z) cartesian coordinates. A cloud of points is one of the multiple representations that can be produced from the sensors raw data, other representations include rectangular meshes, quad-tree representations and triangular grids.[1] Depending on the range of the LIDAR sensor, large areas can be scanned with high detail with a single scan, however, this can create data clouds of up to 70,000 points or more. The range can be reduced to cover only up to 20 or 30 meters and less than a 180 degrees, while angle resolution is manipulated to reproduce a fairly accurate representation of the terrain without unnecessarily increasing the number of points per scan. Figure 3.1 shows a point cloud representation obtained with 3D LIDAR sensing.

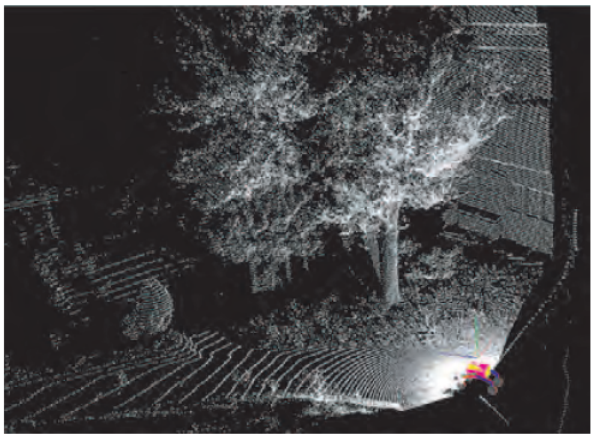


Figure 3.1: Outdoor location scene taken with a LIDAR sensor[1]

3.1.3 Local Map

With the LIDAR data obtained, a representation of the topography of the surface surrounding the mobile rover is defined as a 3D point cloud set of heights, z, corresponding to a coordinate pair (x,y). These (x,y,z) locations are assumed to be expressed in the coordinate frame of a local map. This map representation is commonly known as a Digital Elevation Map (DEM), previously described.

The local DEM, created with LIDAR data, has a proper resolution to represent terrain features on a scale appropriate for rover traversal. This map can be further discretized as a grid, where each cell will be represented as a node on a grid space with defined boundaries, defined as the Local Map. On a simple case, when this Local Map is processed, nodes can be specified as free space or obstacles with corresponding costs for traversal difficulty to be used

by a path planner. Therefore, this local map becomes the workspace on which the vehicle will plan a series of steps to the local goal, and on a higher scale to a global waypoint.

In order to define the Local Map, the measurements of the local DEM are converted to normalized units (i,j,z), such that $i, j \in \mathbb{N}$, by determining the length of the grid cells:

$$\begin{aligned} x &= l_{grid} * i \quad for\ i = 0, 1, ..., N-1 \\ y &= l_{grid} * j \quad for\ j = 0, 1, ..., M-1 \end{aligned} \tag{3.1}$$

Then, let η_{ij} represent a node of the Local Map, where N and M are the total number of nodes on the x and y axes, respectively. An example of a DEM and Local Map representation is given in Figure 3.2, where R_η represents a region of the Local Map centered on η_{ij}.

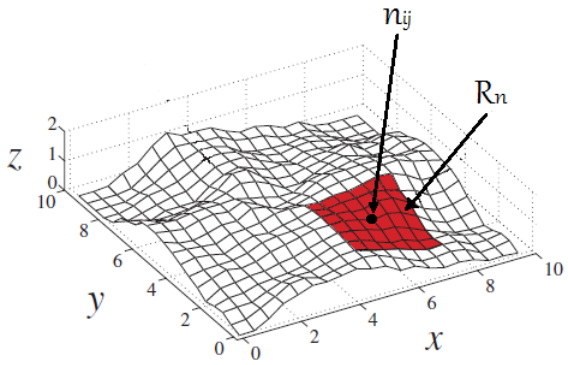


Figure 3.2: DEM and discretized representation of the map[1]

In order to determine the size of the grid, one should consider the resolution that will properly represent map features, however a very high resolution could be unnecessary depending on the size of the rover. Some map representation methods introduce two steps for transforming the terrain map into a traversability map. Initially, small grid cells are created to represent terrain features as obstacles or clear grids. Then, the traversability representation is built using larger grid cells, where each grid cell is assigned the maximum traversability value of any region encompassed by that cell. Then, the grid will be marked untraversable if any part of that cell is an obstacle. However, this again encompasses the same problem, the definition of an initial and secondary grid size to properly represent terrain features, without unnecessarily blocking clear paths.[9]

Chapter 4

Traversability Map Representation

The Traversability Map (TM) is a representation of the natural terrain that classifies difficulty of traversal for a mobile robot over areas, or cells, of the Local Map (LM). The Traversability Map offers a processed map representation that can be used by a path planner, given that each cell has a defined traversability factor. Thus, a cost can be created from this factor and included in the path planner's decision making process in order to avoid obstacles and hazardous regions.

4.1 Binary Traversability Map

The Binary Traversability Map (BTM) is a grid representation of a terrain map, where each node is determined to be either free or non-traversable. Each cell in the LM is represented as a node in the map space χ_{Mspace} , where (i, j) are the normalized (x, y) map positions described in Chapter 3.

In a Binary Map Representation, every map node η_{ij} belongs to one of two sets of the map space, χ_{free} and χ_{obs}, as given by the following relation:

$$\chi_{free} = \{\eta_{ij} \mid (\eta_{ij} \in \chi_{Mspace}) \bigwedge (\eta_{ij} \notin \chi_{obs})\}, \quad 0 \leq i, \leq N-1, \quad 0 \leq j, \leq M-1$$

$$\chi_{obs} = \{\eta_{ij} \mid (\eta_{ij} \in \chi_{Mspace}) \bigwedge (\eta_{ij} \notin \chi_{free})\}, \quad 0 \leq i, j \leq N-1, \quad 0 \leq j, \leq M-1 \tag{4.1}$$

Given the previous definition, there is no intersection of the two subsets, such that $\chi_{free} \bigcap \chi_{obs} = \emptyset$.

The traversability factor corresponding to each set is defined as follows:

$$f_t(\eta_{ij}) \quad = \quad \begin{cases} 0 \quad if \ \eta_{ij} \ \in \chi_{free} \\ \\ 1 \quad if \ \eta_{ij} \ \in \chi_{obs} \end{cases} \tag{4.2}$$

Unknown or occluded terrain can be assumed to belong to one of the two sets, χ_{free} or χ_{obs}. The BTM is used by this work primarily for demonstration purposes, and is therefore defined as a user input.

4.2 Traversability Map Representation for Natural Terrain

The Binary Traversability Map assigns a binary value to each cell, where a 1 is used to denote an obstacle, and a 0 represents an obstacle-free cell. This methodology is simple and commonly used, but it disregards intrinsic terrain properties. In contrast to the Binary Map Representation, the Multi-Valued Traversability Map presented next is a continuous valued local map representation. The aim is to define the characteristics of the natural

terrain in a way that can be integrated into a path planning logic as costs or constraints. Therefore, a local path planner can implement these costs into the decision making process, effectively creating a safe path to traverse. In the Traversability Map Representation, every map node η_{ij} belongs to one of the following sets:

$$\begin{aligned}
\chi_{free} &= \{\eta_{ij} \mid (\eta_{ij} \in \chi_{Mspace}) \bigwedge (\eta_{ij} \notin \chi_{obs} \bigwedge \eta_{ij} \notin \chi_{trav})\}, \quad 0 \leq i, \leq N, \quad 0 \leq j, \leq M \\
\chi_{obs} &= \{\eta_{ij} \mid (\eta_{ij} \in \chi_{Mspace}) \bigwedge (\eta_{ij} \notin \chi_{free} \bigwedge \eta_{ij} \notin \chi_{trav})\}, \quad 0 \leq i, j \leq N, \quad 0 \leq j, \leq M \\
\chi_{trav} &= \{\eta_{ij} \in \chi_{Mspace}) \bigwedge (\eta_{ij} \notin \chi_{free} \bigwedge \eta_{ij} \notin \chi_{obs})\}, \quad 0 \leq i, j \leq N, \quad 0 \leq j, \leq M
\end{aligned} \tag{4.3}$$

with corresponding traversability factor given the set to which a node belongs:

$$f_t(\eta_{ij}) \quad = \quad \begin{cases} 0 \quad if \ \eta_{ij} \ \in \chi_{free} \\ \\ 1 \quad if \ \eta_{ij} \ \in \chi_{obs} \end{cases} \tag{4.4}$$

Otherwise

$$f_t(\eta_{ij}) \in (0, 1) \quad if \ \eta_{ij} \in \chi_{trav} \tag{4.5}$$

4.2.1 Roughness and Inclination Index

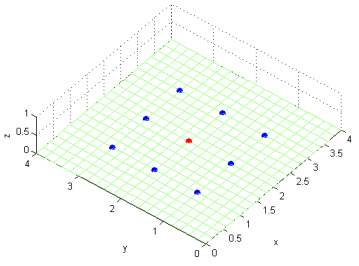
The traversability characteristics of the terrain can be assessed using a roughness, inclination and step index (RIS) to analyze terrain features and characteristics. The RIS index is in turn used to define a traversability factor which will serve as a representation of traversal difficulty for the terrain map, conforming the Traversability Map.

The terrain RIS index, σ_{eta}, of a node η_{ij} with respect to a terrain region is defined as:

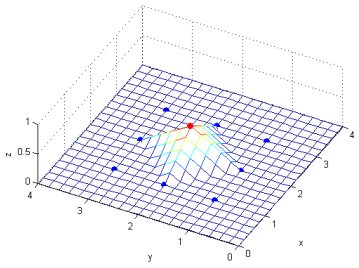
$$\sigma_{\eta} = \sqrt{\frac{1}{n}\sum_{R_{\eta}}[z(R_{\eta}) - z(\eta)]^2} \tag{4.6}$$

where R_{η} is a region of n nodes centered on η_{ij}, $z(R_{\eta})$ is the altitude of a node inside the region, and $z(\eta)$ is the elevation of the node η_{ij}. For an eight-connected grid space, described in Chapter 4, there are 8 nodes neighboring η_{ij}. Therefore, for the scope of this work, $n = 8$, such that the region R_{η} used to compute the RIS index is restricted to the eight neighboring nodes. Figure 4.1 shows the RIS index of the center node, shown in red, for three different cases. Naturally, the index for a planar terrain is zero, and it is higher for a single obstacle (with z=.5) than that of a slight slope.

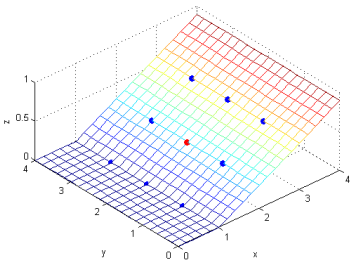
The index can be compared to the work presented in previous investigations.[1, 10] On this work, the RIS index groups three types of terrain characteristics, requiring user analysis only over a single parameter. Moreover, the index is a simple and easily computed parameter that characterizes a single cell node, avoiding the loss of detail on the terrain representation, compared to other methods that utilize parameters to categorize an entire fitted plane or region.



(a) $\sigma_\eta = 0$



(b) $\sigma_\eta = 0.5$



(c) $\sigma_\eta = 0.2598$

Figure 4.1: RIS index evaluation

4.2.2 Traversability Factor

After computing the RIS Index, a threshold value, τ_o, is selected to detect obstacle nodes, as well as the traversability factor of the remaining nodes. The obstacle threshold is user-defined, and depends on an analysis on rover clearance height and pitch hazard. Each cell on the Traversability Map will have a defined factor.

The Traversability factor is defined as follows:

$$f_t(\eta_{ij}) = \begin{cases} 1 & if \quad \sigma(eta) > \tau_o, \quad s.t.\ \eta \in \chi_{obs} \\ 0 & if \quad \sigma(eta) = 0, \quad s.t.\ \eta \in \chi_{free} \\ \sigma(\eta) * \frac{1}{\tau_o} & if \quad 0 < \sigma(eta) < \tau_o, \quad s.t.\ \eta \in \chi_{trav} \end{cases} \tag{4.7}$$

Upon determining the factors, cell nodes that surpass the obstacle threshold will now belong to the obstacle set, χ_{obs}. The following section will explain the consideration taken by the path planner over obstacle cells.

Chapter 5

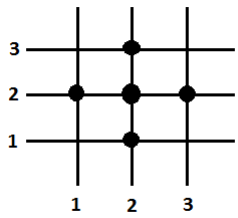
Path Planning

Path planning is segmented in two steps. On the first step, a search algorithm determines a series of node steps to the goal over a grid space, based on minimizing a cost function. A path-relaxation algorithm is implemented as a second step to fine tune the location of path nodes. This chapter focuses on the description of the first path planning step, its methods and implementation.

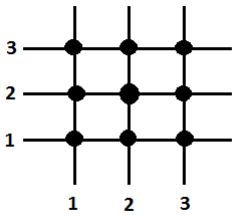
5.1 Search Space

Path planning searches consist of two components: the search space and the search algorithm itself. If concerned with navigation and motion planning, search spaces typically consist of simple primitives that satisfy certain mobility requirements. Cell decomposition methods break a continuous space on which the search algorithm must perform into a finite number of cells, yielding a discrete search problem. A grid is the simplest cell decomposition search space commonly implemented. The optimality of the path may depend on the type of grid motion allowed. Traditionally, motion over the grid space is strictly allowed between the nodes. A four-connected grid, shown in Figure 5.1, blocks

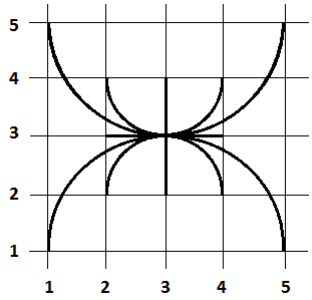
diagonal paths even if there is enough space to move through them. Thus, it may become necessary to reduce the size of the grid in order to allow for more open paths, increasing the number of nodes in the map.



(a) Four connected grid



(b) Eight connected grid



(c) Arc-based primitive

Figure 5.1: Grid Search Space. a) shows the movement in transitions of 90 deg, whereas b), an eight-connected grid, allows diagonal movement as well. A simple arc-based motion is shown on c), where movement is allowed from the node through the arcs.

An eight-connected grid (each node connected to it's diagonal, as well as orthogonal neighbors) eliminates the problem with diagonal paths as motion is allowed in heading transitions of 45 degrees. Figure 5.1 shows the eight-connected grid, where a robot located in node (2,2) can move to any of its eight neighbors.

The eight-connected grid still shows inefficiencies on the trajectory outcome. Instantaneous changes of direction are usually caused by the use of a grid search space; thus, it cannot certainly satisfy smoothness and vehicle dynamic constraints that require minimizing abrupt heading changes. Improved methods include modifying the motion over the grid to an arc-based primitive, which can increase the number of neighboring states that emanate from each node, as seen in Figure 5.1.

Instead of using a more complex grid, an eight-connected grid is implemented and a path-relaxation step is proposed. The path relaxation algorithm is an optimization step that fine-tunes the location of the initial path nodes over a grid space, determined by the first step of path planning, and returns a smooth trajectory. The algorithm is described on Chapter 6 in detail.

5.2 Search Algorithm

The search algorithm used is an iterative improvement algorithm, a Heuristic Hill Climbing Search.[13] Iterative improvement algorithms often provide the most practical approach, however, based on a local search, they

often require enhanced versions for optimality and completeness. Hill Climbing algorithms have all the states of the space laid out on a surface, where the height of any point corresponds to an evaluation function at that point, as observed in Figure 5.2. If the algorithm views the evaluation function as a cost instead of quality, and the intention is to reach a minimum, not a maximum, then it is said to be of a gradient descent class. For the purpose of this work, an initial, unmodified surface cost is represented by the Traversability Map.

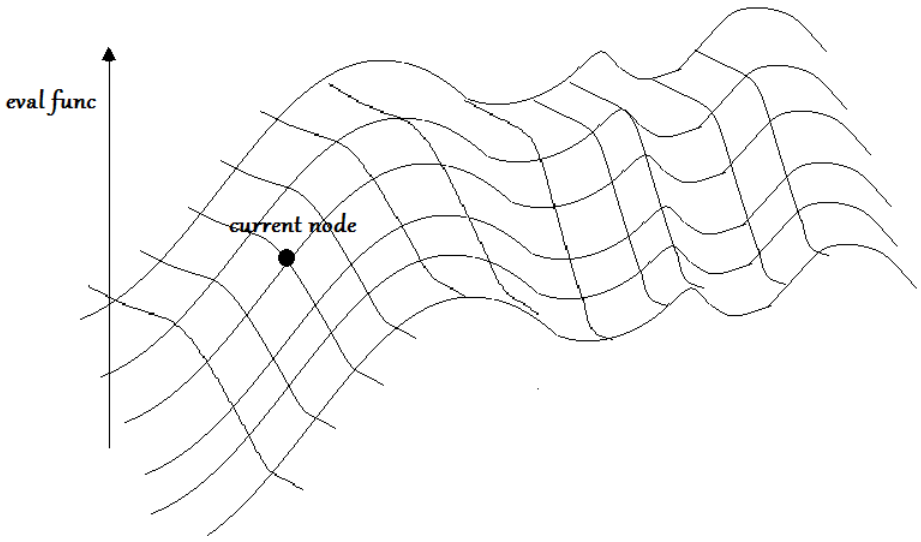


Figure 5.2: Hill climbing algorithm looks for maxima or minima on a surface defined by an evaluation function

Traditionally, hill climbing algorithms keep track of only the current state, and do not look ahead beyond immediate neighbors of that state. The algorithm does not attempt to exhaustively try every node and path, so no node list or array is maintained, just the current state. Due to their nature, traditional Hill climbing algorithms can get stuck on local minima. A heuristic

repair method is used in this work to avoid this and promote movement toward a solution. The same heuristic cost described in Chapter 2 is added to the cost function to attract the sate towards the goal, where

$$h(\eta_{ij}) = \| \eta_g - \eta_{ij} \| \tag{5.1}$$

The cost surface is further modified by manipulating the traversability factors that conform the TM into a traversability cost average, extending the region of local knowledge directly accounted by the Hill Climbing algorithm, and further into a complete cost function, $C(\eta)$, such that η is a vector of neighboring nodes. At each iteration, the algorithm considers the single elements of η and accepts the change that improves $C(\eta)$, such that η is locally optimal.

5.3 Cost Function

The path planning search algorithm's decision making is based on selecting a series of trajectory waypoints leading to the goal that minimize a cost function. The cost is spread from the current location to the neighboring nodes with the following cost function:

$$C(\eta_{ij}) = W_L h(\eta_{ij}) + W_{obs} C_{obs}(\eta_{ij}) + W_v C_v(\eta_{ij}) \tag{5.2}$$

The optimistic heuristic cost $h(\eta_{ij})$ is computed as the distance from the node η_{ij} to the goal node such that $h(\eta_{ij}) = \| \eta_g - \eta_{ij} \|$. This is called

an optimistic cost because it assumes the path from the current node to the goal as a straight line, the minimum cost possible for the given node. If η_{ij} belongs to an array χ_v of previously visited nodes, a cost C_v is added in order to promote the path planner to explore new areas. Moreover, if it is necessary for the rover to retrace its steps to get out of a dead-end, for example, the introduction of this cost does not block the previously visited nodes.

C_{obs} is the Traversability Cost Average value created from incoming traversability factors (f_t) of neighboring nodes taken from the Traversability Map. Nodes in spacious/high-traversability regions have lower values, nodes in densely obstacle populated areas have higher values. The inclusion of the Traversability Cost Average in the cost function allows to choose paths through open areas as a trade off for short length paths. This cost, as well as the algorithm to compute it is explained in the next section. The values W_l, W_{obs}, and W_v are weight values added to the different costs that form the cost function. The addition of these weight values can offer the user to manipulate the decision parameters over the planned trajectory. However, the inclusion of the Traversability Cost Average in the cost function allows to choose paths through open areas as a trade off for short length paths.

5.3.1 Traversability Cost Average

The Traversability cost average defines a cost utilizing information from the TM. Whereas the actual difficulty of traversal for each node is specified

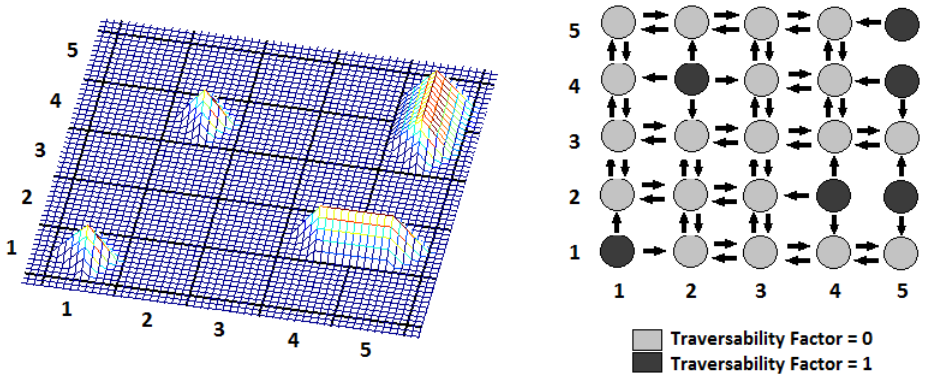
by the traversability factor, the hazard of traversing a topological region is defined by the traversability cost average. For example, a near object or a node difficult to traverse adds to the cost of the node in question. The nearer the obstacle, the more cost it adds to the node. This average is quite helpful when implementing a local search algorithm because it expands the region of knowledge for the decision making process.[16]

Consider a small region of a DEM map shown in Figure 5.3(a) and the cost average representation on Figure 5.3(b). Figure 5.3(b) is represented as a binary traversability map, where the traversability factors of each node are either 0 or 1, free space or an obstacle. Each node receives traversability information from the neighboring nodes, where links are broken if one of them is an obstacle. Obstacle nodes have no incoming links since it is unnecessary to obtain their cost average. This happens because nodes marked as obstacles are blocked; movement to those nodes is not allowed and thus they do not require a cost average.

Each node adds incoming information from neighboring nodes and creates a temporary value, and divides it by 2^1 on 1st iteration, 2^2 on second, etc. This value converges to a steady value after a very few iterations. Then, if greater than zero, the original node value is added to the temporary value. After this is done, nodes in open areas will have lower values than nodes in dense areas. In addition, the cost average allows the cost function to have a trade-off between short path length and lowering the chance of backtracking

due to an unpredictable obstacle.[16] Table 5.1 shows the Traversability Cost Average algorithm.

The region R_η centered on node η_{ij} must be defined again to compute the Cost Average, C_{obs}. This region represents the area of knowledge over which information relevant to the node is considered. Figure 5.3(b) shows a region extended two nodes away from the center node. The number of calculations and therefore processing time increases for a larger R_η. Chapter 7 shows results for a cost average over a region extended 3 nodes away from the center node.



(a) Map with obstacles

(b) Representation of the Traversability Cost Average

Figure 5.3: Computation of the Traversability Cost Average

Table 5.1: Traversability Cost Average Algorithm

Input : $Traversability Map, \quad \eta_{ij},$

Output : $Traversability\ Cost\ Average$

1 : $define\ R_{\eta}$
2 : $create\ \ c_{temp}(R) = 0$
3 : $define\ \ c_t = TM_R$
4 : **for** $i = 1 : 2$
5 : $k = 2^i$
6 : $for\ \eta \in R_{eta} \notin \chi_{obs}$
7 : $for\ p = -1\ to\ 1$
8 : $for\ q = -1\ to\ 1$
9 : $c_{temp\ \eta} = c_{temp\ \eta} + c_{t\ \eta_{pq}}$
10 : **end**
11 : **end**
12 : $c_{temp} = \frac{c_{temp}}{k}$
13 : **end**
14 : $c_t = c_{temp}$
15 : **end**
16 : $C_{obs} = c_t(\eta_{ij})$

Chapter 6

Path Relaxation

6.1 Relaxation

The path planning search algorithm implemented minimizes computation requirements by using a local search over a discretized space. However, the trajectory created by this and other path planning methods traditionally implemented on grid search spaces, most likely, will not be smooth and require sharp turns due to the implementation of an eight-connected grid space. Vehicle dynamic constraints may require minimizing abrupt heading changes,[8] thus the initial path may not be optimal for real life implementation. Also, this grid space definition may increase path length unnecessarily because of a limited motion from node to node. The Path Relaxation step intends to solve these problems.

After the grid search has created an initial path, the path relaxation is an optimization step that fine-tunes the location of each node on the path to minimize the total cost of the trajectory. Each node's position is adjusted in turn using only local information to minimize the cost of the path sections on either side of that node.

In order to avoid grouping of nodes, each node is allowed to move only

one unit on a line perpendicular to a vector intersecting the previous and next trajectory nodes. The cost for node motion depends on distance traveled (from preceding, through current, to next node) and proximity to obstacles and terrain features. It would be difficult to solve for the minimum cost due to node position on a continuous line without knowing an exact relation between the cost and node position. Therefore, a binary search is used to find a minimum to within a given tolerance or resolution.[14]

6.2 Cost Function and Constraints

In order to move the location of each path node $\eta_p = (i_p, j_p)$ on the local map, the slope between the previous and next node, m, is calculated and its negative reciprocal, m_p determined. Now, an optimal location for each node, η_{pi}, can be found on the line given by m_p, using a binary search to determine the position on this line with the minimum cost. To initialize this step, the maximum location on the line, given by $\eta_{max0} = (x_{max0}, y_{max0})$, is one unit distance away from the initial node's location, such that

$$\begin{aligned} x_{max0} &= \sqrt{\frac{1}{m_p^2+1} + x_{pi}} \\ y_{max0} &= \sqrt{\frac{m_p^2}{m_p^2+1} + y_{pi}} \end{aligned} \tag{6.1}$$

and

$$\eta_{min0} = \eta_{pi} \tag{6.2}$$

The cost is determined by local information on path length to minimize the cost of the path sections on either side of that node, as well as proximity to obstacles and features. If any of the neighboring nodes on the quadrant along of the line of motion has an obstacle, then the initial node η_{pi} is not allowed to move. This is in order to avoid any further proximity to an obstacle. The cost function is given by

$$C(\eta_{pi,k}) = f(\eta_{pi,k}) + g(\eta_{pi,k}) + \frac{1}{o(\eta_{pi,k})} \tag{6.3}$$

where

$$f = \|\eta_{pi-1} - \eta_{pi,k}\| \tag{6.4}$$

$$g = \|\eta_{pi+1} - \eta_{pi,k}\| \tag{6.5}$$

$$o = \|\eta_{obs} - \eta_{pi,k}\| \tag{6.6}$$

where η_{obs} is an obstacle node within a region along the quadrant of motion.

The path relaxation step returns the new location of the path's nodes. After a node has been moved (relaxed) then the node's position may no longer be integer valued. Therefore, the path relaxation returns an array

$$\eta_p = (x_p, y_p), \quad \textit{such that } x_p, y_p \in \mathbb{R}^+ \tag{6.7}$$

It would be difficult to solve for the minimum cost due to node position. Therefore, the binary search is used to find a minimum to within a given tolerance or resolution. Three iterations of the binary search already allows for a resolution of $\frac{1}{8}$ along the line of node motion, and thus can be used as a stopping criteria. This fine-tunning binary search does not guarantee the solution to be an optimal point on the line of motion, but it can at least guarantee that each step provides a monotonically decreasing cost for the solution. A second iteration of path relaxation will further fine-tune the location of the nodes on the path and may offer a sufficiently optimized solution. Figure 6.1 shows three iterations of a binary search, where $\eta_i = (x_i, y_i)$ is the initial location of the path node, taken as the initial $\eta_{min} = (x_{min}, y_{min})$ value for node motion. The path relaxation algorithm is shown in Table 6.1

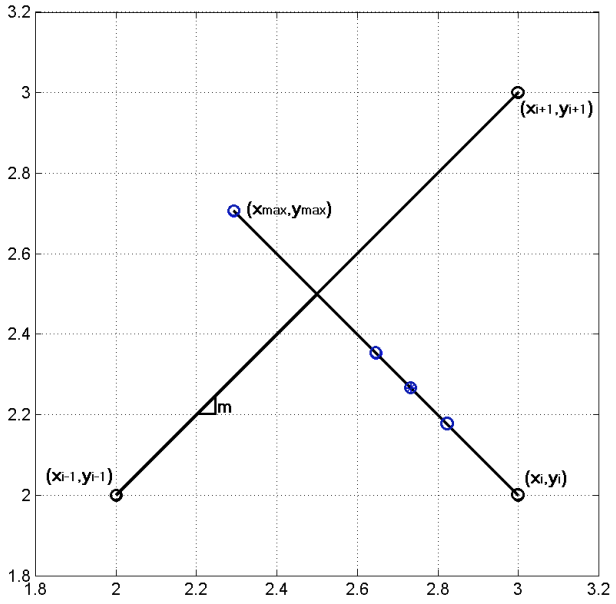


Figure 6.1: Binary search example. Binary search steps are shown as blue points, with the solution after $k = 3$ marked with an asterisk.

Table 6.1: Path Relaxation algorithm

Input : $\eta_p, TM,$

Output : *Relaxed Path*

1 : **for** $j = 1 : 2$
2 : **while** $i \leq path\ length$
3 : $Compute : m_i \rightarrow m_{ip}$
4 : $Compute : \eta_{max0}$
5 : $\eta_{min0} = \eta_{pi}$
7 : $Calculate\ f_{max0}, g_{max0}, o_{max0}$
8 : $f_{min}, g_{min}, o_{min} \rightarrow C_{max}, C_{min}$
5 : **for** $k = 1 : 3$
6 : **Binary Search**
9 : **if** $C_{kmax} > C_{kmin}$
10 : $x_{k+1max} = (x_{kmax} + x_{kmin})/2$
11 : $y_{k+1max} = (y_{kmax} + y_{kmin})/2$
12 : **else if** $C_{kmax} < C_{kmin}$
13 : $x_{k+1min} = (x_{kmax} + x_{kmin})/2$
14 : $y_{k+1min} = (y_{kmax} + y_{kmin})/2$
15 : **end if**
16 : **if** $C_{maxk} > C_{mink}$
17 : $\eta_{pi} = \eta_{kmin}$
18 : $Compute\ C_{max\ k+1}$
18 : **else**
19 : $\eta_{pi} = \eta_{kmax}$
20 : $Compute\ C_{min\ k+1}$
20 : **end if**
21 : **end for**
22 : $i = i + 1$
23 : **end while**
24 : $j = j + 1$
24 : **end for**
25 : $return\ \eta_p$

Chapter 7

Results

The Simulation and Results chapter presented consists of two sections. First, to determine the feasibility of implementing a local search algorithm, a simulation was constructed to perform the experimentation on a simple Binary Traversability Map. This simulation takes as an input the Binary Map, and intends to test the basic concept of implementing a heuristic Hill Climbing search algorithm with the use of a Traversability Cost Average to extend the region of local knowledge. The second section demonstrates the feasibility of the algorithm when implemented on a natural terrain map, where it is important to define the difficulty of traversal of the terrain beyond obstacles and free space with a Multi-Valued Traversability Map. All simulations presented were performed in MATLAB.

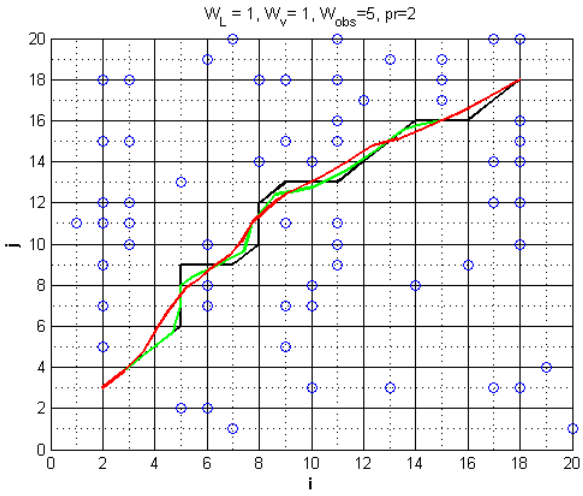
7.1 Simulation Results for a Binary Traversability Map Representation

This section intends to demonstrate the effects of implementing an informed local search algorithm with an extended region knowledge and a heuristic cost over a simple Binary Traversability Map.

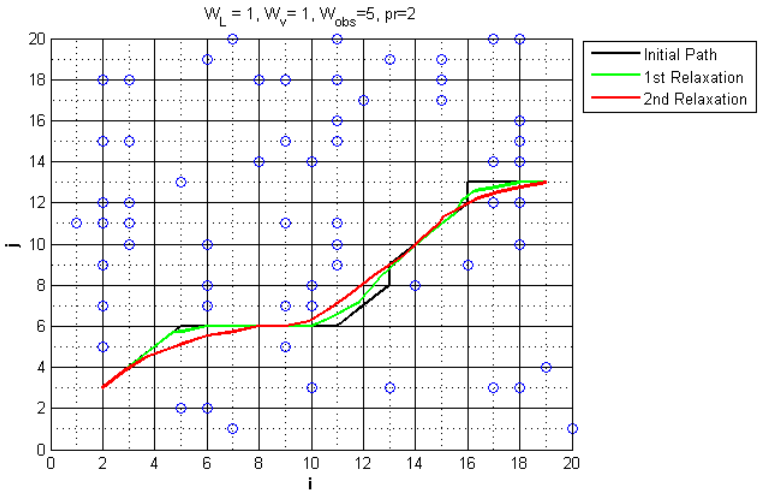
The first examples observed, Figure 7.1(a) and (b) show two paths created over the same Binary TM with different selected goals and fixed weight parameters. For both cases, the initial location of the path is positioned on the lower left corner. Obstacles are denoted by blue dots, and the original path created by the Heuristic Hill Climbing algorithm is shown in black. Two path relaxation steps are implemented, the first one shown in green, and the second step in red. Table 7.1 shows the simulation parameters defined for these results. We can observe that the original path is sub-optimal with respect to length, an ill-condition caused by the characteristics of the search space. As mentioned before, an eight-connected regular grid allows only for transitions of 45 degrees between nodes. The problem becomes present when there is a clear path in a free space, however the planner cannot create a straight line due to the characteristics of the search space. Thus, these examples evidence the practicality of relaxing the initial trajectory. Not only does this accomplish minimizing the cost of traversal, but now the relaxed path follows a smooth trajectory without any drastic heading changes.

7.1.1 Parameter Effects

The previous results have provided a visual introduction of the implementation of the path planning algorithm over a BTM. The goal now is to determine the effects of the parameters that must be selected by the user over a larger map. Figures 7.2 and 7.3 show two different paths created from the same start node to the same goal, with given inputs and parameters defined



(a) Map with $goal = (18, 18)$



(b) Map with $goal = (19, 13)$

Figure 7.1: Relaxed Path, Binary Traversability Map representation over a small 20x20 unit region

Table 7.1: Simulation Parameters

Inputs	**Weight Parameters**
Start: (2,3) Goal:	$W_{obs} = 5$ $W_L = 1$ $W_v = 1$
Simulation Parameters Map Size: 20 x 20 Obstacles: 60 R_η : 7x7 nodes Path Relaxation: 2 iterations	

in Table 7.2. The difference in the paths generated is an effect of selecting different weight parameters for the Hill Climbing search cost function. Figure 7.2 shows the path created giving W_L and W_{obs} an equal weight. This path tries to cut cost by shortening the length of the trajectory, finding a path through obstacle dense areas, and compared to Figure 7.3, it seems to follow a path closer to a straight line to the j value of the goal node. Figure 7.3 shows the alternate path created by selecting a higher value for W_{obs}. Here, the path passes through more open areas, and does not have to backtrack as in the previous map.

Both Figures show that the relaxed path follows the original path closely when in dense areas, though it still smooths out sharp turns. Moreover, the relaxed path on Figure 7.2 discards the backtracking step of the original path through the bottom left corner of the map.

The previous results demonstrate optimizing effects of weight parame-

Table 7.2: Simulation Parameters

Inputs	**Weight Parameters**
Start: (4,4) Goal: (45,35)	W_{obs} W_L W_v
Simulation Parameters Map Size: 50 x 50 Obstacles: R_η : 7x7 nodes Path Relaxation: 2 iterations	

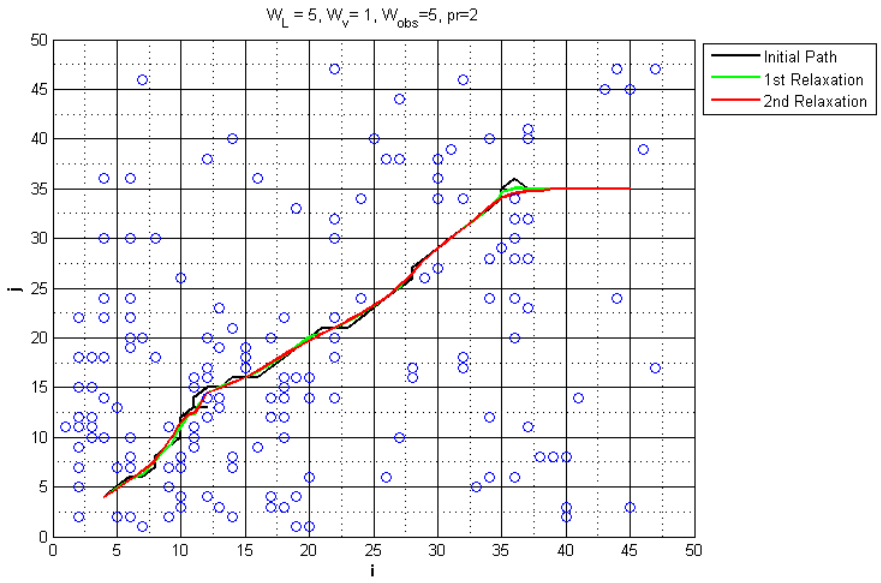


Figure 7.2: Relaxed path, Binary Traversability Map

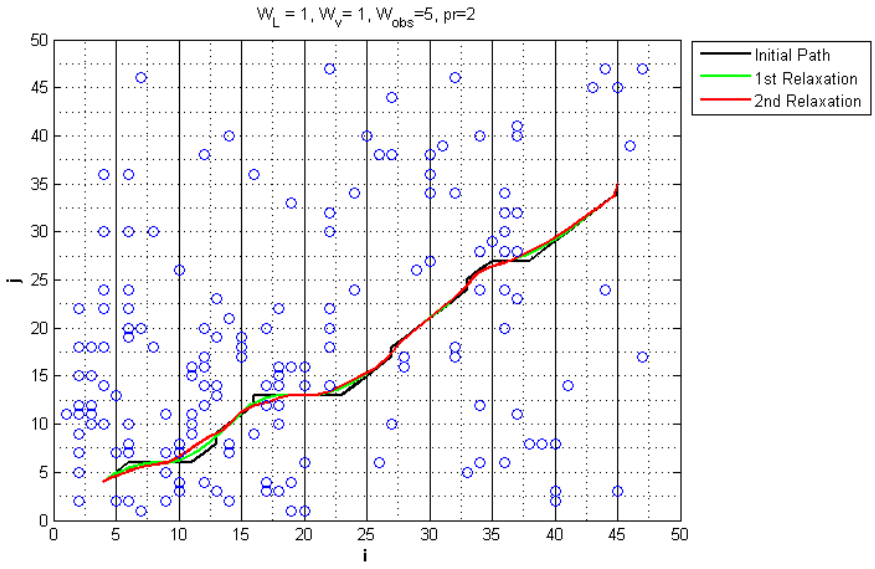


Figure 7.3: Relaxed path, Binary Traversability Map with Weight effects

ters and path relaxation over specific examples. In order to analyze the effects on time and path length for both the original and relaxed path planning, Tables 7.3-7.5 show a numerical analysis over a series of weight parameters for a small, medium, and large map. PL_1 defines the path length of the original path created by the hill climbing search algorithm, while PL_2 gives the length of the relaxed path after 2 iterations. The time t_1 gives the processing time for the original path and t_2 is the total processing time after including the two relaxation steps, both given in seconds.

Results shown in these tables are averaged over 100 runs, where the location of obstacles on the Binary Map are input for each run as random integers given by a discrete uniform distribution. Results show that for the

three cases, the original path has a shorter length for a smaller weight ratio ($W_L > W_{obs}$). This is to be expected considering that a higher ratio causes the path to avoid obstacle-dense areas, an attempt that may cause the path to deviate from a straight line as necessary.

As observed in Figures 7.2 and 7.3, the relaxed path follows the original path closely in obstacle-dense areas, such that reduction of path length in such areas may be negligible. Numerical results shown in Tables 7.3-7.5 show that the percentage decrease in path length is much lower for low weight ratios, implying again that a higher W_{obs} avoids dense areas, allowing for more open space for the path to be relaxed. This in turn results in a higher path length decrease. Moreover, both t_1 and t_2 seem relatively unaffected by changes in the weight ratio.

Table 7.3: Path Length and Time vs. Map Size and Parameter Ratio, Map Size= 25x25

Map Size = 25x25						
	$\frac{W_{obs}}{W_L} = 1$	$\frac{W_{obs}}{W_L} = 2$	$\frac{W_{obs}}{W_L} = 5$	$\frac{W_{obs}}{W_L} = 10$	$\frac{W_{obs}}{W_L} = \frac{1}{5}$	$\frac{W_{obs}}{W_L} = \frac{1}{2}$
PL_1	22.012	22.91	22.508	22.51	21.846	22.294
PL_2	21.536	21.588	21.558	21.378	21.404	21.56
t_1	0.00342	0.00428	0.00350	0.002925	0.00292	.00316
t_2	0.0763	0.07762	0.07775	0.07568	0.07432	.07624
$(1 - \frac{PL_2}{PL_1}) * 100$	4.543	5.902	4.221	5.029	2.033	3.2924

Table 7.4: Path Length and Time vs. Map Size and Parameter Ratio, Map Size= 50x50

Map Size = 50x50						
	$\frac{W_{obs}}{W_L} = 1$	$\frac{W_{obs}}{W_L} = 2$	$\frac{W_{obs}}{W_L} = 5$	$\frac{W_{obs}}{W_L} = 10$	$\frac{W_{obs}}{W_L} = \frac{1}{5}$	$\frac{W_{obs}}{W_L} = \frac{1}{2}$
PL_1	44.16	44.728	44.064	44.228	42.426	43.188
PL_2	41.518	41.766	41.934	41.92	41.278	41.288
t_1	.00745	0.00836	.0075	.00748	.00752	0.00884
t_2	.0807	0.08196	.0817	.08416	.0795	.0827
$(1 - \frac{PL_2}{PL_1}) * 100$	5.9828	6.623	4.834	5.22	2.706	4.3994

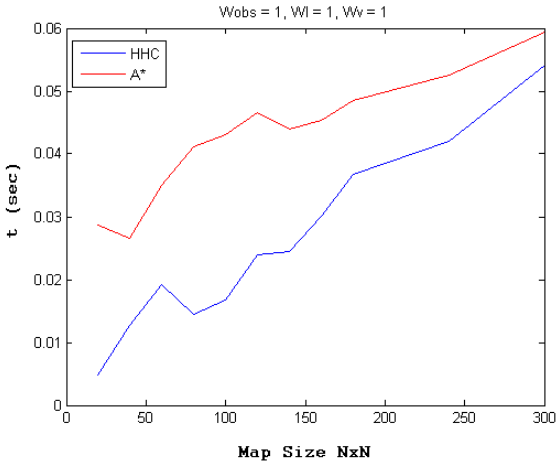
Table 7.5: Path Length and Time vs. Map Size and Parameter Ratio, Map Size= 100x100

Map Size = 100x100						
	$\frac{W_{obs}}{W_L} = 1$	$\frac{W_{obs}}{W_L} = 2$	$\frac{W_{obs}}{W_L} = 5$	$\frac{W_{obs}}{W_L} = 10$	$\frac{W_{obs}}{W_L} = \frac{1}{5}$	$\frac{W_{obs}}{W_L} = \frac{1}{2}$
PL_1	99.49	98.496	100.74	99.586	97.998	99.06
PL_2	96.72	96.712	96.648	97.282	96.486	96.502
t_1	0.01564	0.01586	0.01578	0.0163	0.01776	0.01572
t_2	0.09034	.0883	0.0908	0.09044	0.0912	.08892
$(1 - \frac{PL_2}{PL_1}) * 100$	2.7842	1.8133	4.062	2.314	1.543	2.59

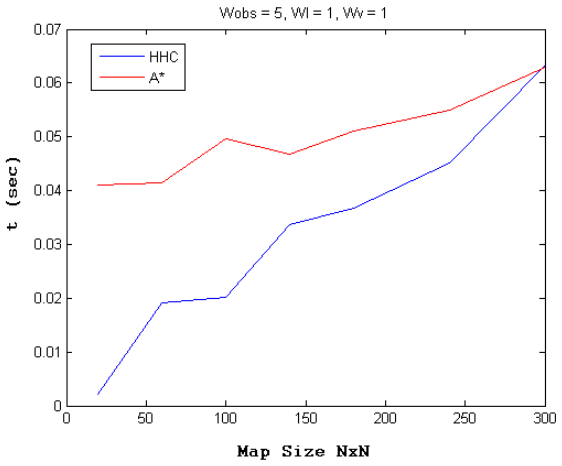
7.1.2 Numerical Comparison with Traditional A*

A comparison analysis to verify the HCC's improved efficiency over the A* algorithm for trajectory planning may be complex as several performance criteria can be defined. However, the objective of this work is to propose a local search algorithm that improves the usage of computational resources while providing a safe path to the goal. The efficiency of the HHC will be judged by a numerical analysis of its improvement of processing time, and a few examples will be shown to visually demonstrate effectiveness in comparison to the traditionally implemented A*.

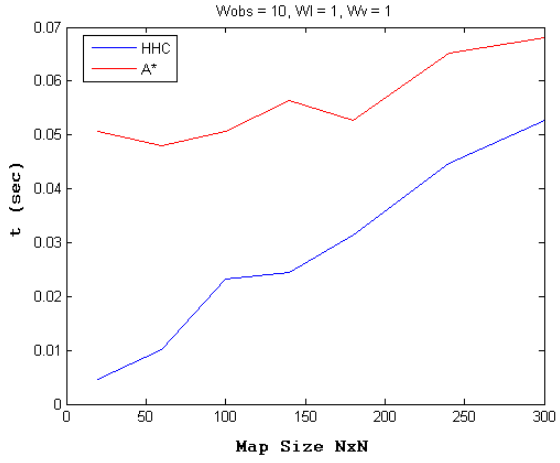
Figure 7.4 shows the processing time in seconds vs. map size for the A* algorithm using a binary heap and the Heuristic Hill Climbing search algorithm, with parameters given in Table 7.6. The results of this numerical comparison show excellent performance of the HHC algorithm regarding processing time for local maps of up to 300x300 nodes. The HHC algorithm performs considerably faster for maps where $N < 250$ throughout the different weight parameters. Assuming that a LIDAR sensor gives a higher resolution at a shorter range, it may be desired to limit the size of the local map to approximately 50 m. Given a reasonable grid length, the size of the Local Map may not go over 200x200 nodes. Therefore, these results are quite relevant for the proposed scenario.



(a) $\frac{W_{obs}}{W_L} = 1$



(b) $\frac{W_{obs}}{W_L} = 5$



(c) $\frac{W_{obs}}{W_L} = 10$

Figure 7.4: Processing time comparison for HHC and A* path planning algorithms

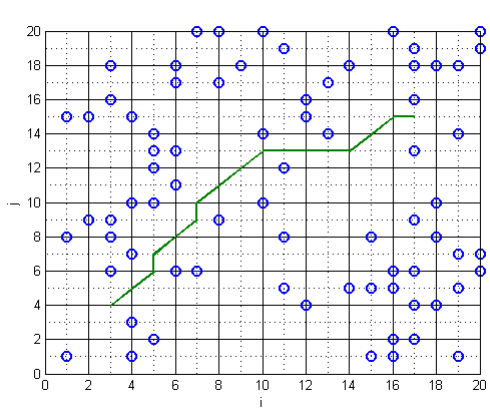
Table 7.6: Simulation Parameters

Inputs	**Weight Parameters**
Start: (3,N/2) Goal: (N-4,N/2)	$W_{obs} = 1, 5, 10$ $W_L = 1$ $W_v = 1$
Simulation Parameters Map Size: NxN Obstacles: 2*N R_η : 7x7 nodes Path Relaxation: Off	

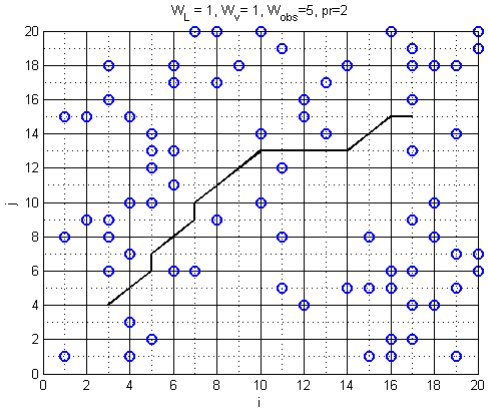
Next, Figures 7.5 and 7.6 show examples of a small sized and medium sized map, respectively, and the trajectories generated by A* and HHC algorithms. It would be complex to analyze comparative trajectories with respect to performance criteria such as hazard avoidance or feasibility of the path in a general manner. These maps offer an analysis only over specific cases, but are a visual aid to the numerical results previously presented.

Figure 7.5 shows a small sized map, with equal paths generated by the A* and HHC algorithms. The processing time for the path generated by the HHC local search algorithm was 10 times faster than that of the A* algorithm, showing excellent performance and accuracy of the HHC algorithm for implementation on small maps.

Figure 7.6 shows two alternative paths generated by the algorithms over a medium sized map. The path on the right seems, at simple sight, to avoid tighter spaces in certain regions. Both paths equally complete the selected task, however for safety purposes, the second one can be considered

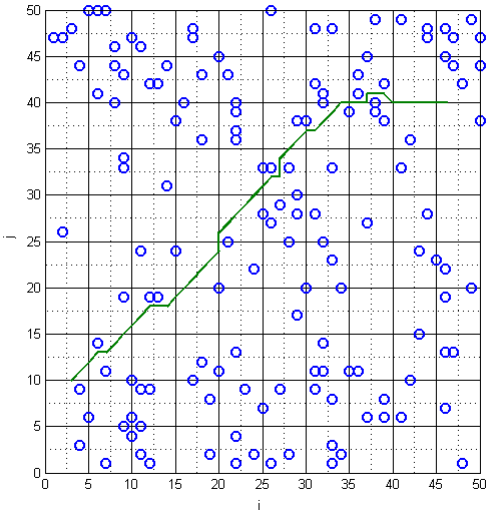


(a) A* generated trajectory, $t = 0.0261sec$

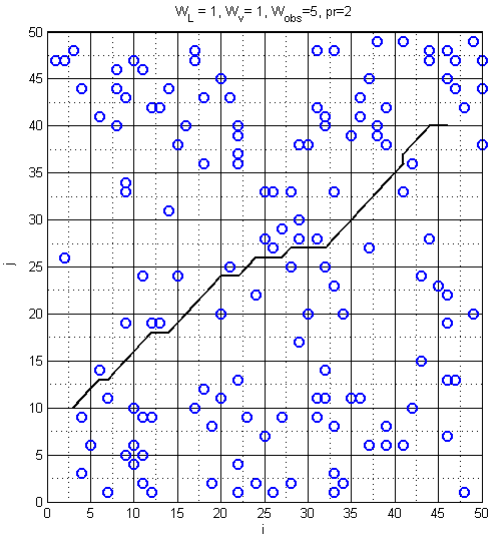


(b) HHC generated trajectory, 0.0025

Figure 7.5: Processing time comparison for HHC and A* path planning algorithms



(a) A*-generated trajectory, $t = 0.0367$ sec



(b) HHC-generated trajectory, $t = .0.0143$

Figure 7.6: Processing time comparison for HHC and A* path planning algorithms

more efficient. The path generated with the HHC algorithm has a processing time $t = 0.0143$ and length $PL = 55.006$, compared to the time $t = 0.0367$ and $PL = 52.764$ generated by A*.

7.2 Simulation Results for a Traversability Map Representation of Natural Terrain

Given the previous results and analysis of the implementation the HHC local search algorithm, this section demonstrates the feasibility of the algorithm when implemented on a natural terrain map, where it is important to define the difficulty of traversal of the terrain beyond obstacles and free space. Table 7.7 shows the inputs and parameters defined for this simulation.

The Path Planning architecture for natural terrain creates the Multi-Valued Traversability Map automatically. It does so by defining the RIS index for each cell of the local DEM, and determining the Traversability Factor. Figure 7.7 shows the DEM map used for the simulation. The map is 20X20 meters, and it represents the perceived local map, as explained in Chapter 3. Figures 7.8 and 7.9 show the RIS Index values for the normalized local map and the Multi-Valued TM, respectively. Given the *obstacle threshold*, τ_o, nodes with a high RIS index are marked as obstacles, as seen in Figure 7.9. The value of τ_o is entirely selected by the user, and it depends upon analysis on the rover's clearance height and pitch hazard. For example, the obstacles seen in the DEM map, Figure 7.7, have a height of 0.5 m, and are considered hazardous obstacles for a small sized rover.

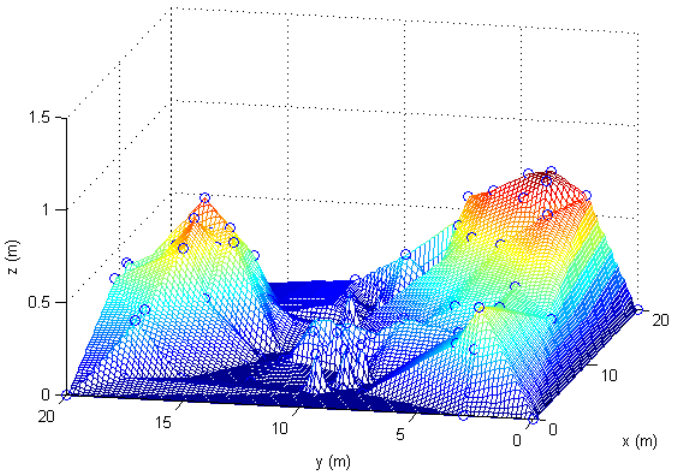


Figure 7.7: Digital Elevation Map

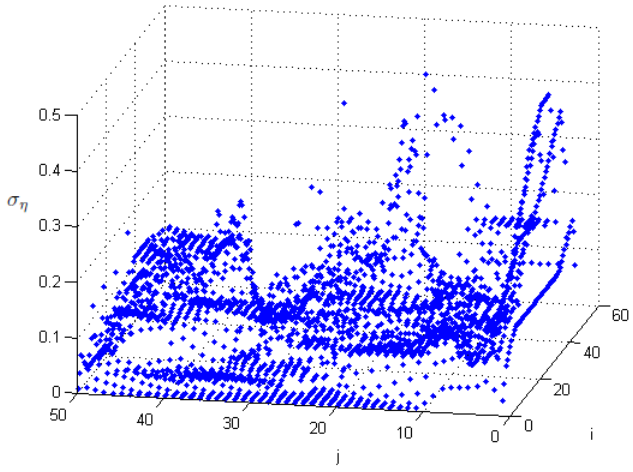


Figure 7.8: RIS Index

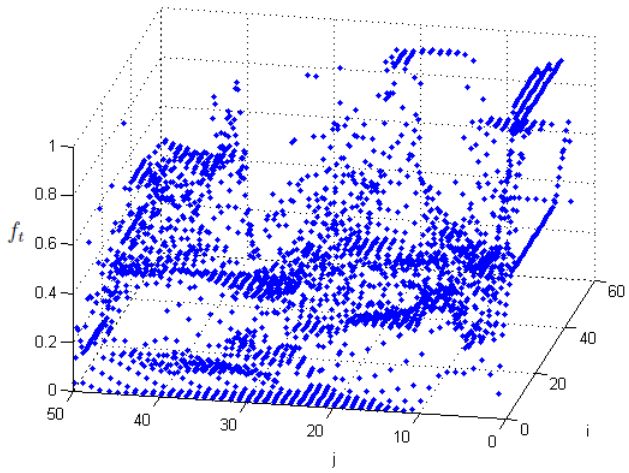


Figure 7.9: Multi-Valued Traversability Map with Traversability Factors

7.2.1 Parameter effects

The goal now is to determine the effects of the parameters, selected by the user, over a Multi-Valued Traversability Map. The weights added to the cost function, W_L and W_{obs}, allow the user to manipulate the decision parameters for path planning. The inclusion of the Traversability Cost Average allows to choose paths through open areas as a tradeoff for short path lengths. The weight given to it determines the importance given to avoidance of hazardous regions, which for the multi-valued case, consist not only of obstacles but rough terrain and areas with slopes.

Figures 7.10, 7.11, and 7.12 shows a surface projection of the Multi-Valued Traversability Map, with the initial path and the relaxed path for two relaxation steps. Nodes marked as obstacles are shown in red. Figure 7.10

Table 7.7: Simulation Parameters

Inputs	**Weight Parameters**
Start: (7,20) Goal: (45,35)	$W_{obs} = 0, 1, 5$ $W_L = 1, 2$ $W_v = 1$
Simulation Parameters Map Size: NxN Obstacles: R_η : 7x7 nodes Path Relaxation: Off	

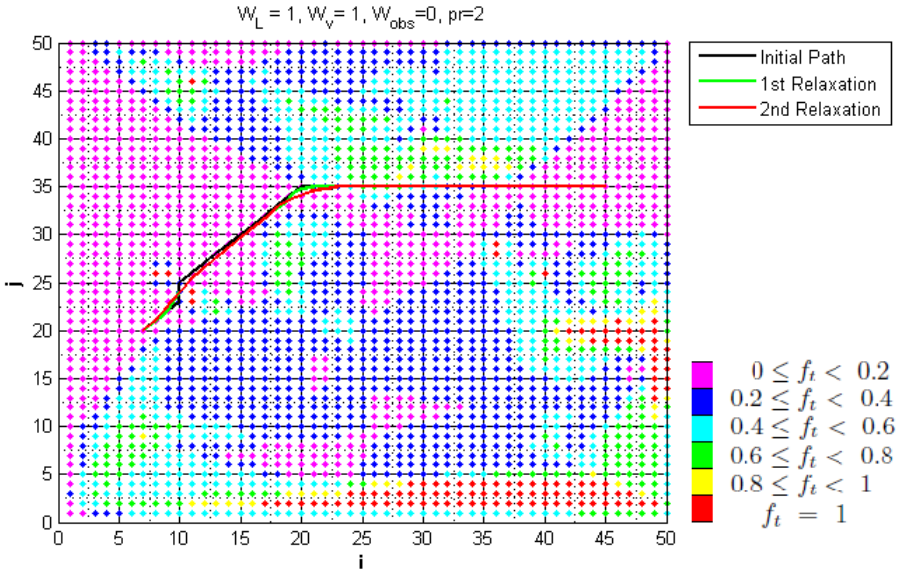


Figure 7.10: Relaxed Path, Multi-Valued Traversability Map, no Obstacle Weight

shows the trajectory selected with an obstacle weight, W_{obs}, of 0. Therefore, the Traversability Cost Average is not taken into account when minimizing the cost function. The selected path passes through areas of nodes with high traversal difficulty, prioritizing path length in the search of the goal node.

Figures 7.11 and 7.12 show the effect of the weight parameters on the path. Choosing equal values for W_L and W_{obs} begins to pull the path away from the hazardous areas and into regions with low hazard values. We can observe the path curving as the gap between the two weights grows in Figure 7.12, and finally, Figure 7.13 shows the trajectory created giving a higher priority to the obstacle cost. Here we see that the trajectory curves to stay within the low cost areas. The path planner has selected a longer trajectory, but has a larger clearance from hazardous areas, which is as expected by prioritizing the Traversability Cost Average.

If the terrain surface can be perceived with high resolution and easily represented as a DEM, as it has been demonstrated by the use of LIDAR technology, then these results demonstrate that the Path Planning algorithm presented can be implemented with confidence over a Multi-Valued Traversability Map, and moreover, over rough terrain. Figure 7.14 shows the final relaxed path resulting from the simulation conditions, transformed to position coordinates on the local DEM.

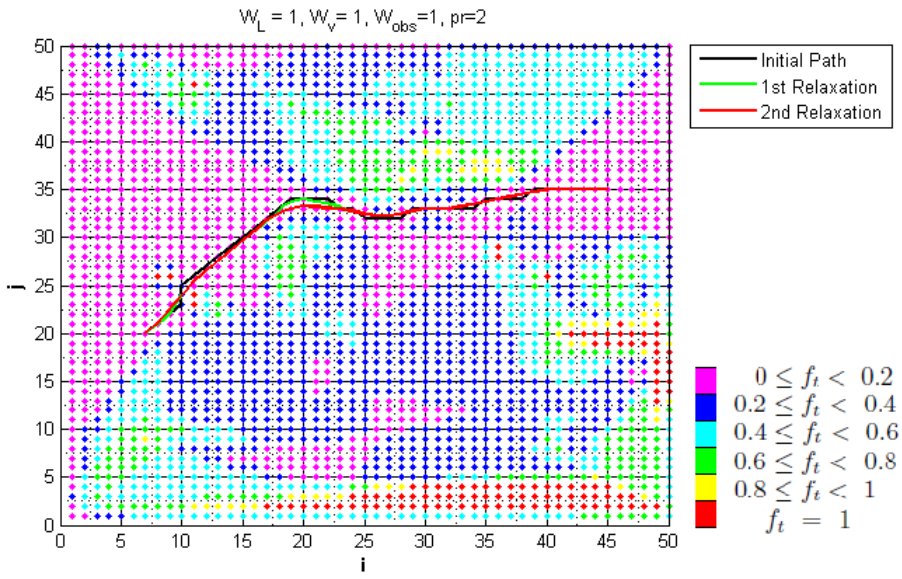


Figure 7.11: Relaxed Path, Multi-Valued Traversability Map

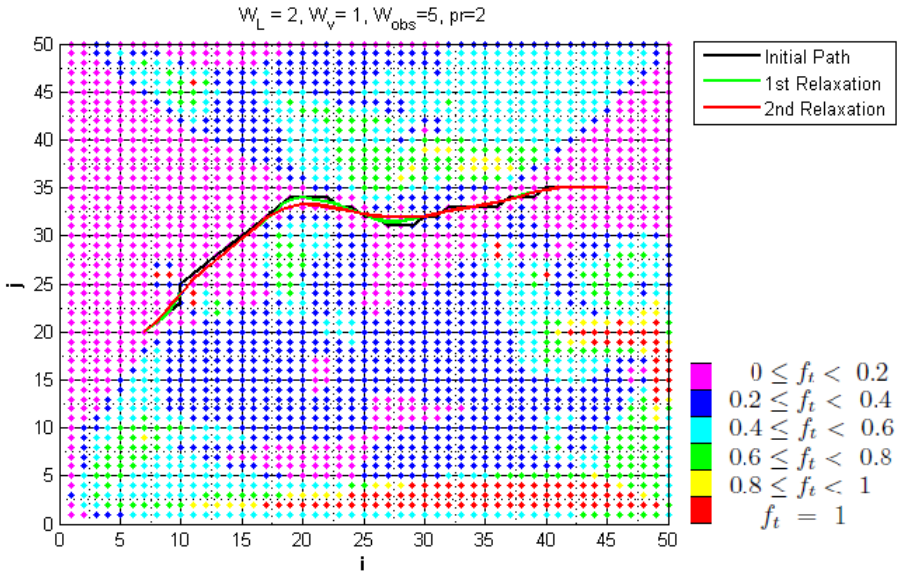


Figure 7.12: Relaxed Path, Multi-Valued Traversability Map with Weight effects

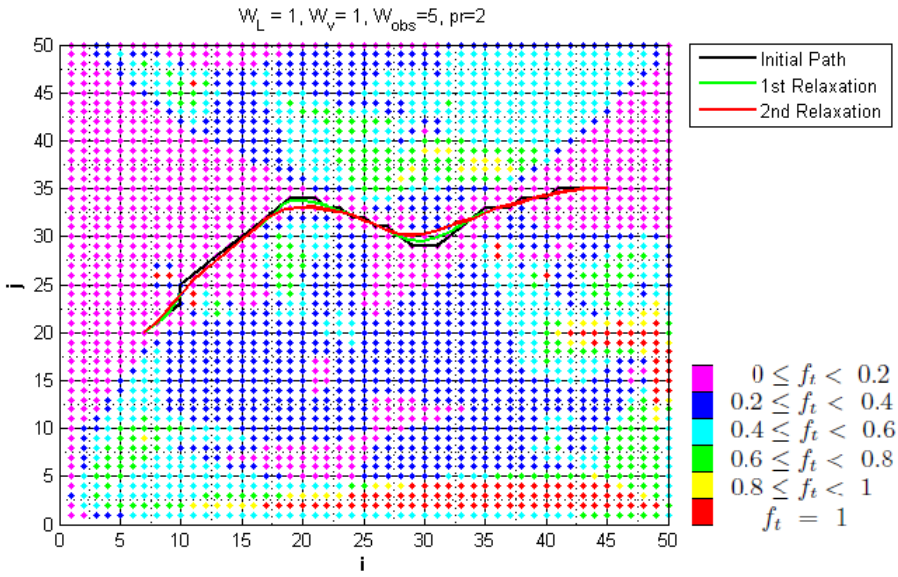


Figure 7.13: Relaxed, Multi-Valued Traversability Map, W_{obs} effects

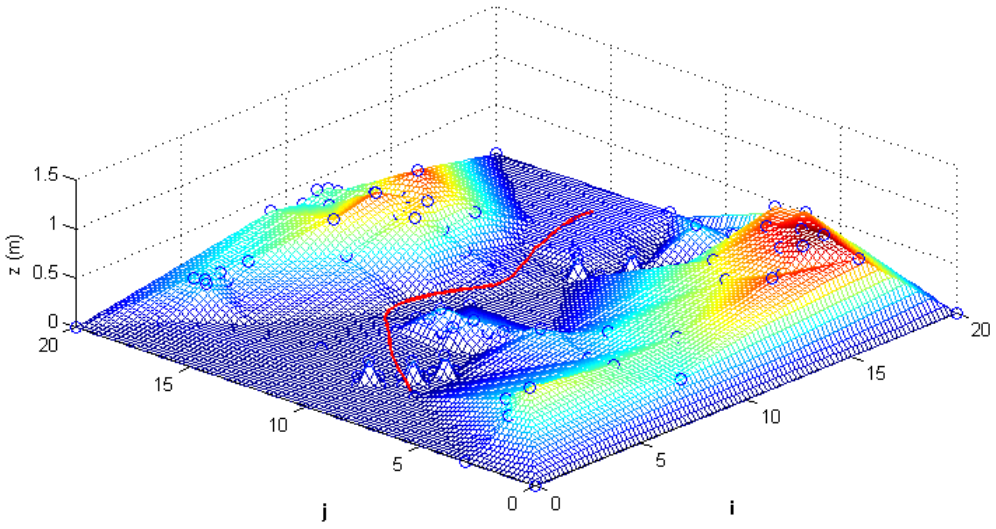


Figure 7.14: Relaxed path over DEM

Chapter 8

Conclusions

This thesis considers the problem of implementing a path planning algorithm that is capable of adding the avoidance of hazardous areas as a priority over short distance paths. Two cases are considered: 1) the terrain grid map is assigned binary valued traversability factors, 2) The terrain grid map is autonomously assigned multi-valued traversability factors.

Traditional approaches have so far implemented an A* search algorithm, extending a global search through the local map to create a path to the goal. This approach, though optimal and complete, is impractical for real life applications due to the exhaustion of every possibility from a node of origin. The heuristic hill climbing search algorithm, in contrary to the traditional approach, may in theory produce a suboptimal solution because it extends a local search, rather than a global one. However, results demonstrate that the introduction of an extended version of this local search algorithm with a traversability cost average has an improved optimality.

The extended heuristic hill climbing algorithm, implemented as the core of the path planning system offers a reduction in processing times for

maps up to 300x300 nodes, as compared to the traditional A* implementation. This represents a realistic advance for practical implementation with LIDAR mapping, where it is desirable to restrict range to optimize resolution. These results would therefore be quite generous with respect to map size.

Reducing processing time and requirements of the search algorithm without reducing optimality allows the implementation of other technologies that enhance autonomous path planning. For example, the development of an automatic multi-valued traversability matrix that can acknowledge intrinsic terrain properties enhances navigation over natural terrain. Similarly, a path relaxation algorithm that smoothes the planned trajectory complies with control requirements on heading changes. These developments benefit from a search algorithm with a lower usage of computation resources, and altogether increase the autonomy of an exploration vehicle.

The automated processing of the digital elevation map and obtention of the multi-valued traversability factors, for example, add processing time and memory usage to the path planning, independent of the search algorithm implemented. However, results demonstrate that the enhanced hill climbing algorithm has good performance over the presented multi-valued traversability matrix, avoiding hazardous regions rather than only single obstacles. A more extensive study should be performed for future research to reduce overall computing requirements of this presented architecture.

On the other hand the path relaxation algorithm, which is presented as an independent and easy to implement optimization step that fine tunes the location of each node of the trajectory, intends to smooth the path without adding complexity and processing requirements on the search algorithm. Future research should compare and numerically investigate the benefits of path relaxation versus increased search space complexity over computing resources.

Bibliography

[1] A. Mora, K. Nagatani, and K. Yoshida. Path planning and execution for planetary exploration rovers based on 3d mapping. In *Mobile Robots Navigation*, pages 263–288. InTech, 2010.

[2] J.J. Biesiadecki, P.C. Leger, and M.W. Maimone. Tradeoffs between directed and autonomous driving on the mars exploration rovers. *The International Journal of Robotics Research*, 26(1):91–104, 2007.

[3] K. Gushwa, A. Guerrero de la Peña, J. Cervantes, C. Patel, and W. Fowler. Dorra the europa explorer: An orbital and in-situ investigation of habitability. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2011.

[4] B.P Malay. *Stellar-aided Inertial Navigation Systems on the Surface of Mars*. PhD thesis, University of Texas at Austin, 2003.

[5] R. Volpe. Rover functional autonomy development for the mars mobile science laboratory. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 2, pages 643–652. IEEE, 2003.

[6] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz. Global path planning on board the mars exploration rovers. In *Proceedings of the 2007 IEEE Aerospace Conference*, pages 1–11, 2007.

[7] E. Gat, M.G. Slack, D.P. Miller, and R.J. Firby. Path planning and execution monitoring for a planetary rover. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, volume 1, pages 20–25, 1990.

[8] M. Pivtoraiko, T.M. Howard, I. Nesnas, and A. Kelly. Field experiments in rover navigation via model-based trajectory generation and nonholonomic motion planning in state lattices. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*. IEEE, 2008.

[9] A. Howard, H. Seraji, and B. Werger. A terrain-based path planning method for mobile robots. In *Proceedings of the 7th International Conference on Automation Technology*, 2003.

[10] S.B. Goldberg, M.W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proceedings of the 2002 IEEE Aerospace Conference*, volume 5, pages 2025–2036, 2002.

[11] D.M. Helmick, A. Angelova, M. Livianu, and L.H. Matthies. Terrain adaptive navigation for mars rovers. In *Proceedings of the 2007 IEEE Aerospace Conference*, pages 1–11, 2007.

[12] I. Rekleitis, J.L. Bedwani, and E. Dupuis. Over-the-horizon, autonomous navigation for planetary exploration. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2248–2255, 2007.

[13] S.J. Russell, P. Norvig, J.F. Candy, J.M. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*. Prentice hall, 2010.

[14] C. Thorpe and L. Matthies. Path relaxation: Path planning for a mobile robot. In *OCEANS*, pages 576–581. IEEE, 1984.

[15] P.J.F. Carle, P.T. Furgale, and T.D. Barfoot. Long-range rover localization by matching lidar scans to orbital elevation maps. *Journal of Field Robotics*, 27(3):344–370, May 2010.

[16] M.G. Slack and D.P. Miller. *Path planning through time and space in dynamic domains*. 1987. Technical Report TR-87-05, Computer Science, Virginia Polytechnic Institute and State University.

[17] A. Lacaze, Y. Moscovitz, N. Declaris, and K. Murphy. Path planning for autonomous vehicles driving over rough terrain. In *Proceedings of the IEEE ISIC/CIRA/ISAS Joint Conference*, pages 50–55, 1998.

Vita

Ana Isabel Guerrero de la Pena was born in Torreon, Coahuila, Mexico on April 22, 1986. She began her studies at Monterrey Tech studying Mechatronics Engineering, and later received a Bachelor of Science in Aerospace Engineering from the University of Texas at Austin. Upon graduation, she continued her studies at UT Austin in August of 2009. Her research interests include guidance and control of planetary exploration rovers.

Permanent contact: anaisaguerrerop@gmail.com

Permanent address: 2718 Williamsburg Ct
Columbus, IN 47203

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.